Securing Apache Web Traffic with SSL

1. Objectives

The objective of this lab is to perform a quick guide to generating and installing an SSL certificate on an Apache web server with the mod_ssl module.

Normal web traffic is sent unencrypted over the Internet. That is, anyone with access to the right tools can snoop all of that traffic. Obviously, this can lead to problems, especially where security and privacy is necessary, such as in credit card data and bank transactions. The Secure Socket Layer is used to encrypt the data stream between the web server and the web client (the browser).

SSL makes use of what is known as asymmetric cryptography, commonly referred to as public key cryptography (PKI). With public key cryptography, two keys are created, one public, one private. Anything encrypted with either key can only be decrypted with its corresponding key. Thus if a message or data stream were encrypted with the server's private key, it can be decrypted only using its corresponding public key, ensuring that the data only could have come from the server.

2. Lab Tasks:

1. Configuring Apache Server to use SSL
2. Modify the included SSL Apache Virtual Host file to point to our generated SSL certificates.
3. Modify the unencrypted Virtual Host file to automatically redirect requests to the encrypted Virtual Host.

3. Lab Environment

Prerequisites – Digital Signatures and PKI – SI Lab

Before you begin, you should have some configuration already taken care of. In this lab, we need two things, which are already installed in the provided VM image:

(1) the Firefox web browser
(2) the Apache web server

If you don't already have that up and running, you can quickly fix that by typing:

```
# apt-get update
# apt-get install apache2
```

a) Starting the Apache Server.

The Apache web server is also included in the pre-built Ubuntu image. However, the web server is not started by default. You need to first start the web server using the following command:

```
# service apache2 start
```

The above URLs are only accessible from inside of the virtual machine, because we have modified the `/etc/hosts` file to map the domain name of each URL to the virtual machine's local IP address (`127.0.0.1`). You may map any domain name to a particular IP address using `/etc/hosts`. For example you can map `http://www.pkisilab.com` to the local IP address by appending the following entry to `/etc/hosts`:

```
127.0.0.1 www.pkisilab.com
```

If your web server and browser are running on two different machines, you need to modify `/etc/hosts` on the browser's machine accordingly to map these domain names to the web server's IP address, not to `127.0.0.1`.

b) Configuring Apache Server to use SSL

We have created our key and certificate files under the **`/home/silabs/ca`** directory. Now we just need to modify our Apache configuration to take advantage of these.

Enable SSL Module

```
# a2enmod ssl
```

`'default-ssl'` can be replaced by the real site name you set up in `/etc/apache2/sites-available/`. Once the site listed in the command above is enabled with that command, the site will appear in **/etc/apache2/sites-enabled**.

Apply SSL Module to Site

```
#  a2ensite default-ssl
#  /etc/init.d/apache2 restart
```

In the pre-built VM image, we use Apache server to host all the web sites used in the lab. The name-based virtual hosting feature in Apache could be used to host several web sites (or URLs) on the same machine. A configuration file named default in the directory `"/etc/apache2/sites-available"` contains the necessary directives for the configuration:

1. The directive `"NameVirtualHost *"` instructs the web server to use all IP addresses in the machine (some machines may have multiple IP addresses).

2. Each web site has a VirtualHost block that specifies the URL for the web site and directory in the file system that contains the sources for the web site. For example, to configure a web site with URL `http://www.example1.com` with sources in directory `/var/www/Example_1/`, and to

configure a web site with URL `http://www.example2.com` with sources in directory `/var/www/Example_2/`, we use the following blocks:

```
<VirtualHost*>
ServerName http://www.example1.com
DocumentRoot /var/www/Example_1/
</VirtualHost>
```

Edit `/etc/apache2/sites-available/default.` We will set the normal things we'd want to adjust in a Virtual Host file (ServerAdmin email address, ServerName, etc.), adjust the SSL directives to point to our certificate and key files, and uncomment one section that provides compatibility for older browsers.

After making these changes, your server block should look similar to this:

```
    <VirtualHost *:443>
       ServerAdmin   admin@mta.ca
       ServerName    www.pkisilab.com:443
    .
    .
    .
       SSLEngine on
       SSLCertificateFile /home/silabs/ca/server.crt
       SSLCertificateKeyFile /home/silabs/ca/server.key
    </VirtualHost>
```

Save and close the file when you are finished.

c) **Restart your Apache server**

Run the following command:

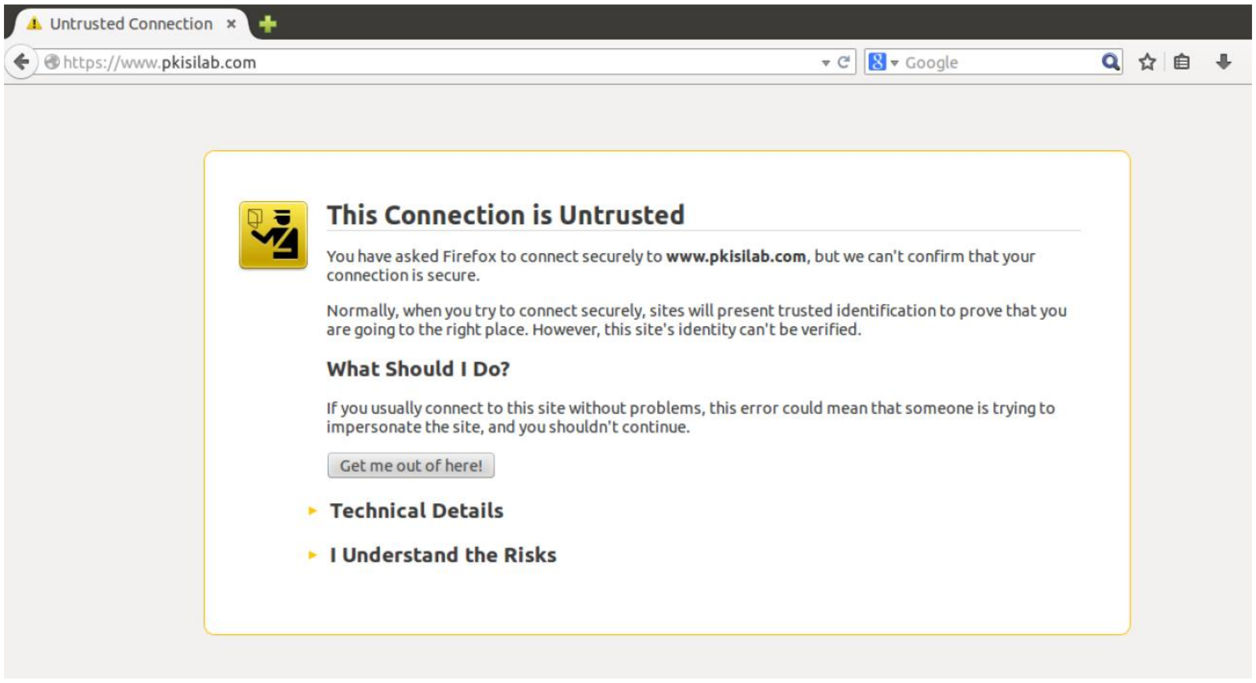
```
# service apache2 start
```

Now, we're ready to test our SSL server.

Open your web browser and type `https://` followed by your server's domain name or IP into the address bar:

```
https://server_domain_or_IP
```

Because the certificate we created isn't signed by one of your browser's trusted certificate authorities, you will likely see a scary looking warning like the one below:



This Connection is Untrusted

You have asked Firefox to connect securely to **www.pkisilab.com**, but we can't confirm that your connection is secure.

Normally, when you try to connect securely, sites will present trusted identification to prove that you are going to the right place. However, this site's identity can't be verified.

What Should I Do?

If you usually connect to this site without problems, this error could mean that someone is trying to impersonate the site, and you shouldn't continue.

Congratulations!!!!

You have configured your Apache server to use strong encryption for client connections. This will allow you serve requests securely, and will prevent outside parties from reading your traffic.

To do:

1. Disable SSL and re-enable it
2. Create a new Virtual Host and customize it for a second domain.
3. As it stands now, the server will provide both unencrypted HTTP and encrypted HTTPS traffic. For better security, it is recommended in most cases to redirect HTTP to HTTPS automatically. Check whether the redirect functions correctly:

```
http://server_domain_or_IP
```